

通过构造 Rop 链绕过 DEP+Aslr

WrittenBy 東

实验环境:windows 7 with sp1 & IE8.0

实验工具:Immunity Debug,mona.py

还是我们之前的这个漏洞,问题是出现在 msxml3.dll 中,之前我们的利用姿势是通过浏览器触发漏洞,当时是在 IE6 触发的漏洞,在新版本 IE8 之后微软提供了一种保护机制 DEP,上节我们也通过 Rop 链绕过了 DEP(数据执行保护)的安全机制,这次的实验环境是 Windows7 这时候又有一个安全机制叫做 Aslr (Address space layout randomization) 地址空间配置随机加载它主要对以下四类地址进行随机化:

1. 堆地址的随机化
2. 栈基址的随机化
3. PE 文件映像基址的随机化
4. 进程环境块 (PEB) 地址的随机化

他在 Vista, Windows2008, Windows7 下是默认启用的 (ie7 除外)

非系统镜像可以通过连接选项/DYNAMICBASE (Visual Studio2005 sp1 以上版本启用该保护)

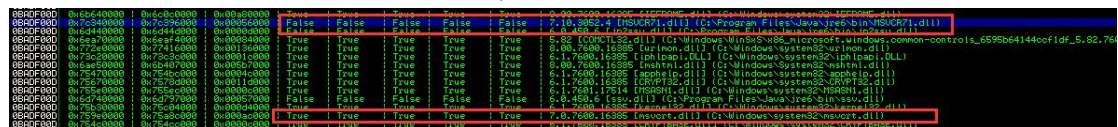
常见绕过 Aslr 的方法:

1. 利用 Aslr 只对高地址随机的特性,覆盖低 16 位部分返回值,进而接管程序流程
2. 利用未开启 Aslr 机制的模块地址,因为这些模块加载时机总是固定的特性无视 Aslr
3. 利用 Heapspray 大面积命中指定位置,只要一个地址有效就可以获得程序执行权限
4. 利用内存泄露,通过内存泄露的信息,比如说指针,可以帮我们构建出完整的内存图谱
5. 利用 SystemCall 驱动级别的底层函数,最开始加载,故地址是恒定不变借此构建特定 api

驱动由于 Aslr 的引入,所有 api 地址都被随机化了,栈的起始位置也随机化了,因此无法通过 Ret2Libc 方式来进行 Payload 了

所以今天我们利用的方法就是通过 Rop 技术利用未开启 Aslr 的模块来绕过 DEP+Aslr

我们可以使用未开启 Aslr 模块的 IAT 来稳定的间接获取目标 API 这也就解释了我们上节的 Rop 链的构造,由于是以二级指针传递的方式存在,因此我们才有了类似 `Jump [eax]` 或者 `call [eax]` 的指令完成对指定函数的调用,而这次在 Win7 下我们可以看到



msvcrt.dll 是默认开启了随机机制的所以我们今天就用 msvcrt71.dll 这样的三无模块在这个 dll 中搜刮我们需要的指令从而构造 Rop 链。

[这个 dll 是 jre.6 下的 dll 安装 Java 后重启后打开浏览器会自行加载的 dll]

指定模块查找指令,先构造我们的 StackPivot 需要 `ret; echg eax,esp ret; pop ebp ret`

0x0c0c28	VirtualProtect Address	
0x0c0c2c	Retn to Payload	
0x0c0c30	lp Address	} 函数栈帧
0x0c0c34	dWsize	
0x0c0c38	flnewProtect	
0x0c0c3C	lpold Protect	
0x0c0c40	Payload..	

这样就形成了一个函数栈帧,最终模拟执行 VirtualProtect

